Sum of Separable Functions in Machine Learning

Jochen Garcke [*]
Technische Universität Berlin
Institut für Mathematik, Sekretariat MA 3-3
Straße des 17. Juni 136, 10623 Berlin, Germany
`garcke@math.tu-berlin.de`

1 Sum of Separable Functions in Machine Learning

We consider the basic binary classification problem, where one starts from a set of labeled data,

$$D = \left\{(\underline{x}_j, y_j)\right\}_{j=1}^{N} = \left\{(x_1^j, \cdots, x_d^j; y_j)\right\}_{j=1}^{N}, \tag{1}$$

with $y_i \in \{-1, 1\}$ labeling the two classes and $\underline{x}$ a d-dimensional feature vector. There exist numerous algorithms for classification (see e.g. [3, 7]). Function based methods for classifying data construct a function $g(\underline{x})$ such that the sign of $g(\underline{x}_j)$ matches y_j for the given data, and the sign of $g(\underline{x})$ correctly predicts y for other $\underline{x}$. Since the data may contain errors, or may simply not provide enough information, one cannot expect to completely satisfy this goal, so one uses suitable loss functions with the aim to minimise the classification error rates.

An approach using functions in high dimensions has to address the curse of dimensionality, where the complexity of the function representation, which is the number of unknowns, typically grows exponentially with the dimensions. One of the reasons for the success of support vector machines is the use of a data centered function representation using kernels. Here the dimensionality turns up in the complexity of the kernel computation and not in the complexity of the function representation which essentially only depends on the number of data.

In the last years numerical approaches using function representations based on a sparse tensor product approach were succesfully used in high dimensions [2, 4]. These approaches are based, in an abstract fashion, on the approximation by a sum of separable functions,

$$g(\underline{x}) = \sum_{l=1}^{r} s_l \prod_{i=1}^{d} g_i^l(x_i), \tag{2}$$

which is very closely related to low rank tensor decomposition, see [8] for a review and further references. The number of terms r is called the *(separation) rank.* Note that the coefficients s_l are solely for later convenience, so that one has $\|g_i^l\| = 1$.

Following [1] we use sums of separable functions of the form (2) but without constraints such as orthogonality or positivity on the g_i^l. The resulting nonlinear approximation method is called a *separated representation* [2]. The functions g_i^l may be constrained to a *subspace*, but are not restricted to come from a particular *basis set.* This extra freedom allows one to find good approximations with surprisingly small r, and reveals a much richer structure than one would believe beforehand. Although there are at present no useful theorems on the size r needed for a general class of functions, there are examples where removing constraints produces expansions that are *exponentially* more efficient than one would expect a priori, i.e. $r = d$ instead of 2^d or $r = \log d$ instead of d, examples are discussed in detail in [2].

[*]based on joint work with Gregory Beylkin and Martin Mohlenkamp

We use a multi-scale basis of tent functions for the g_i^l but one could choose any function space of (finite) dimension M, e.g., polynomials of some degree, splines, wavelets, etc. as well. This space may be different for each term l in the sum, each attribute i, and in general also for each (l, i) pair.

To use the method for classification suitable loss functions need to be used. We employ log-likelihood estimation, which in the sense of probability estimation is more appropriate and statistically sound for classification than a least squares approach [3, 7], and the hinge loss used for support vector machines in a smooth variant introduced in [5].

Since these loss functions are non-quadratic we need to minimise using non-linear minimisation algorithms. We did experiments with the quasi-Newton method BFGS, a non-linear CG-method, and a trust-region method (see e.g. [11]). We investigated both an alternating minimising procedure and a global minimisation which optimises in all dimensions simultaneously, but in our experiments the alternating minimisation procedure achieves much better results than the global minimisation procedure, so we concentrate on the alternating procedure. We expect that with a detailed investigation of the minimisation problem and an adaption of the algorithm to the specific problem the global minimisation will perform better.

In any case we need to formulate a suitable regularised problem. This is necessary to avoid overfitting, but also for numerical stabilisation. Since we employ multi-scale linear functions as our basis for $g(\underline{x})$ we can only use first derivatives in the regularisation. We use $\|\nabla g(\underline{x})\|^2$ as a simple regularisation term, denoted by ∇dD in the following. As an alternative we also investigate the regularisation of each $g_i^l(x)$ by $\|\nabla(g_i^l(x))\|^2$. Due to the orthogonality of the basis we have $\int {\phi_k^l}'(x){\phi_{\tilde{k}}^l}'(x)dx = \delta_{k\tilde{k}}\gamma_k^l$, where γ_k^l depends on the size of the support of ϕ_k^l. We use ∇g_i^l to indicate this regularisation (which is more like numerical stabilisation).

An investigation of the alternating procedure shows that the complexity is

$$\mathcal{O}(K(dr^2M^2 + drMN)S). \tag{3}$$

for the ∇g_i^l regularisation, here we denote the number of BFGS iterations by S and the number of alternating iterations is K. The cost is linear in both d and N, and so the method is feasible for large data sets in high dimensions.

Using the regularisation ∇dD we observe a complexity of

$$\mathcal{O}(K[(dr^2M^2 + drMN)S + d^3r^2M^2]). \tag{4}$$

Again linear in N, but in parts cubic in d, although the inner non-linear solver is linear in d. Nevertheless, the complexity still suggests the method for large data sets in high dimensions.

We compare against results from the study [9], where several classification methods were benchmarked. The separation rank r, the discretisation level of the multi-scale basis (which correlates with the basis size M) and the size of the regularisation parameter were selected using cross-validation. Note that depending on the problem we used up to level 4 of the multi-scale basis and rank $r = 7$, although often $r \leq 4$ was sufficient.

The simple local regularisation method performs slightly better than the global regularisation. For the twonorm, bupa liver and credit data set it results in smaller misclassification rates, while for the spirals and threenorm data set the global regularisation method is better. In particular for some of the real data the minimisation procedure had problems to converge for the global regularisation; again, better adapted non-linear solution strategies might improve the results. The good performance of the simpler local regularisation is an indication that the limitation to a discrete function representation has a large effect in the avoidance of overfitting, known as regularisation by projection in other fields [6, 10].

In comparison to the 17 other methods our approach achieves very competitive results, here we look at the median as is done in [9]. For all data sets at least one variant of our approach is in the top five and for eight of the data sets we are in the top three. For six data sets, the majority, at least one version of our procedure achieved better results than a support vector machine, which was the best method in the referenced study [9].

Table 1: Results on synthetic data from the study [9]. We give the mean (with standard deviation) and median (with inter-quartile range) for the best results from [9], our approach, and our position in comparison to all 17 approaches used. We use log likelihood estimation (L_L), the huberised hinge loss (L_H) and both forms of regularisation.

				$L_L, \nabla g_i^l$		$L_H, \nabla g_i^l$		$L_L, \nabla dD$		$L_H, \nabla dD$	
data set		best other		error	pos	error	pos	error	pos	error	pos
	mean	2.66	svm	2.26	1	2.39	1	**2.22**	1	2.45	1
CIRCLE				(0.44)		(0.44)		(0.46)		(0.47)	
	median	2.50	svm	2.00	1	2.10	1	**1.95**	1	2.30	1
		(0.49)		(0.41)		(0.62)		(0.61)		(0.80)	
	mean	0.17	nn	1.04	3	1.19	3	**0.25**	2	1.03	3
SPIRALS				(0.18)		(0.22)		(0.10)		(0.22)	
	median	0.10	nn	0.90	3	1.10	3	**0.20**	2	0.75	3
		(0.07)		(0.22)		(0.31)		(0.16)		(0.29)	
	mean	2.82	svm	**3.61**	5	4.08	5	12.04	17	6.37	12
TWONORM				(0.34)		(0.34)		(7.85)		(1.34)	
	median	2.70	svm	**3.40**	5	3.85	5	5.50	10	5.90	11
		(0.20)		(0.40)		(0.54)		(8.93)		(0.59)	
	mean	14.17	lvq	18.94	9	19.21	9	**14.45**	2	15.62	2
THREENORM				(0.98)		(0.60)		(0.40)		(0.69)	
	median	13.70	lvq	18.95	8	19.10	9	**14.40**	2	15.40	2
		(0.77)		(0.98)		(0.50)		(0.79)		(1.22)	
	mean	3.58	svm	5.44	2	5.92	2	**4.85**	2	5.43	2
RINGNORM				(0.58)		(2.64)		(0.31)		(0.32)	
	median	2.90	svm	4.80	2	4.90	2	**4.70**	2	5.30	2
		(0.70)		(0.75)		(0.61)		(0.33)		(0.31)	

References

[1] G. Beylkin, J. Garcke, and M. J. Mohlenkamp. Multivariate regression and machine learning with sums of separable functions. *SIAM Journal on Scientific Computing*, 31(3):1840–1857, 2009.

[2] G. Beylkin and M. J. Mohlenkamp. Algorithms for numerical analysis in high dimensions. *SIAM J. Sci. Comput.*, 26(6):2133–2159, July 2005.

[3] C. M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[4] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 2004.

[5] O. Chapelle. Training a support vector machine in the primal. *Neural Computation*, 19(5):1155–1178, 2007.

[6] H. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems.* Kluwer, 1996.

[7] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[8] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, September 2009.

[9] D. Meyer, F. Leisch, and K. Hornik. The support vector machine under test. *Neurocomputing*, 55:169–186, 2003.

[10] F. Natterer. Regularisierung schlecht gestellter Probleme durch Projektionsverfahren. *Numer. Math.*, 28:329–341, 1977.

[11] J. Nocedal and S. J. Wright. *Numerical optimization. 2nd ed.* Springer, 2006.